Silencec – POS project

Nikolas Begetis – nmpegetis@gmail.com

Description

Design and develop the front-end component of a simple Point Of Sales system (POS) where shop tellers can prepare orders and summarize the charges to the client. The system must also send all completed orders to the back-end server for record keeping.

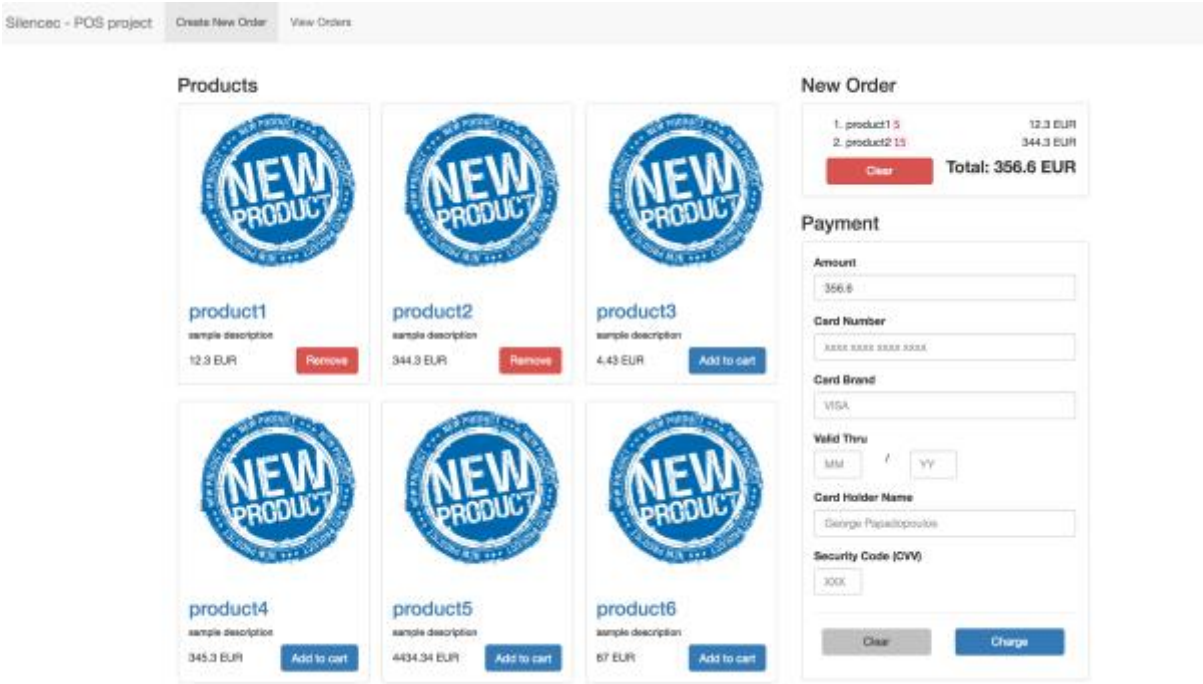

Figure 1: Screenshot from the POS system

Domain Requirements

I decided to create the POS system *(Figure 1)* using the React framework with Redux, ducks and ES6.

I chose **React** because it is the ideal framework for single-page application, and this POS system project matches this description. Also, React supports ES6 and is widely used from the developers' community. React is backed and supported by Facebook while also the active and vibrant community. What is more, JSX is not a problem for me and writing everything in JavaScript doesn't bother me.
What is more, some additional reasons I turned down Angular2 and VueJS against to React, are the following:

1. Angular2: Even though in the past I have worked with Angular and it is a great framework backed from Google, in the updated version Angular2 the developers made some breaking changes in the API which destroyed some of the developers' trust and still Angular2 hasn't got the developer fans that Angular1 had, at least for the time being. So, I decided to put Angular2 aside for the moment.

2. VueJS: Even though it seems to exist a trend in VueJS and a lot of developers support it, it still hasn't got the market share that React and Angular2 have, and is not supported by a lot of big companies yet. Having that in mind and, because I was going to develop this hands-on project, I preferred to do the POS project in a more popular framework, like React, in order to have a more hand-on experience to show, as there are offered more job opportunities where React is requested (except of course I am the lucky one that you hire ☺).
3. React is my first step before also trying React Native, which is also a whole new world. So, that being considered, it helped me with my decision.

So after deciding to use React with ES6, I searched the web for best modern practices in using React and this is how I came up with the **Redux** framework, whose creators designed it having in mind mostly how React was working and what they could do to improve the way React's components connect to each other.

The below two images show how Redux is used and how it simplifies the way everything in a React application is interconnected. The first figure (Figure 2) shows the main idea which is to use a main store where all components bind, store and retrieve their data from.

The second figure (Figure 3) shows a practice of structuring the code using Redux in the layers: Store → Providers → Containers → Actions → Reducers → Store.
Finally, I used the specialization of ducks which suggest bundling together actionTypes & actions & reducers because most of times there exist only one action/reducer pair and there is no need to have them in separate files.

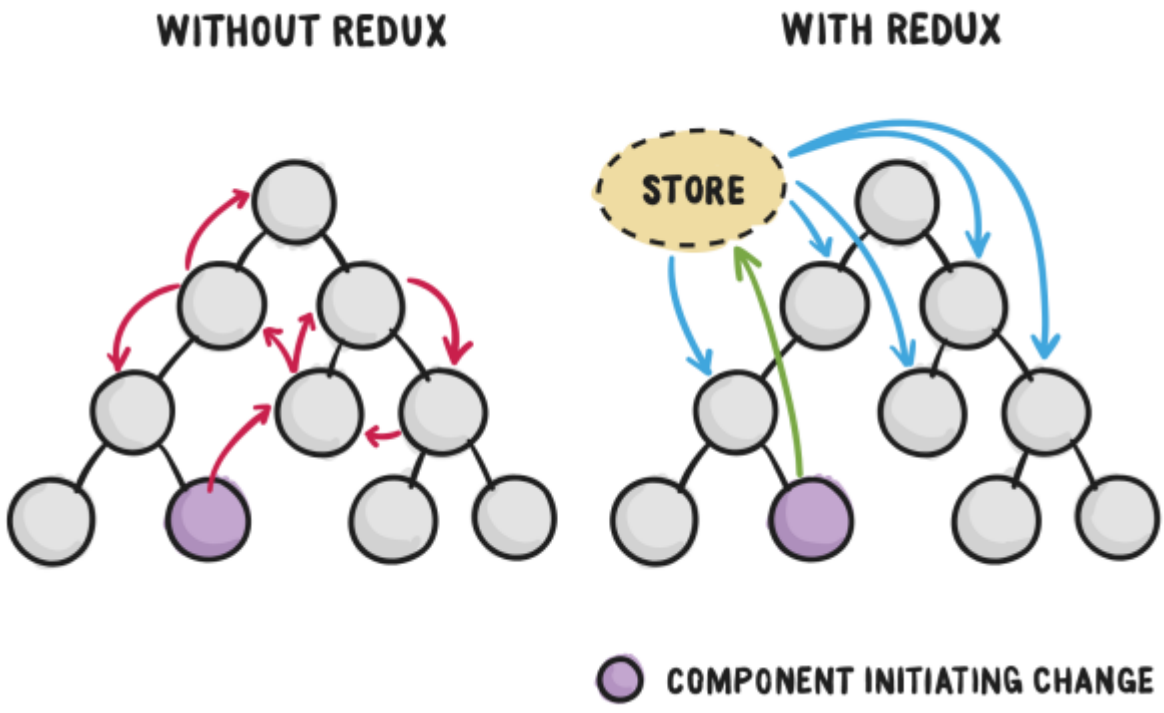

Figure 2: Redux main idea link

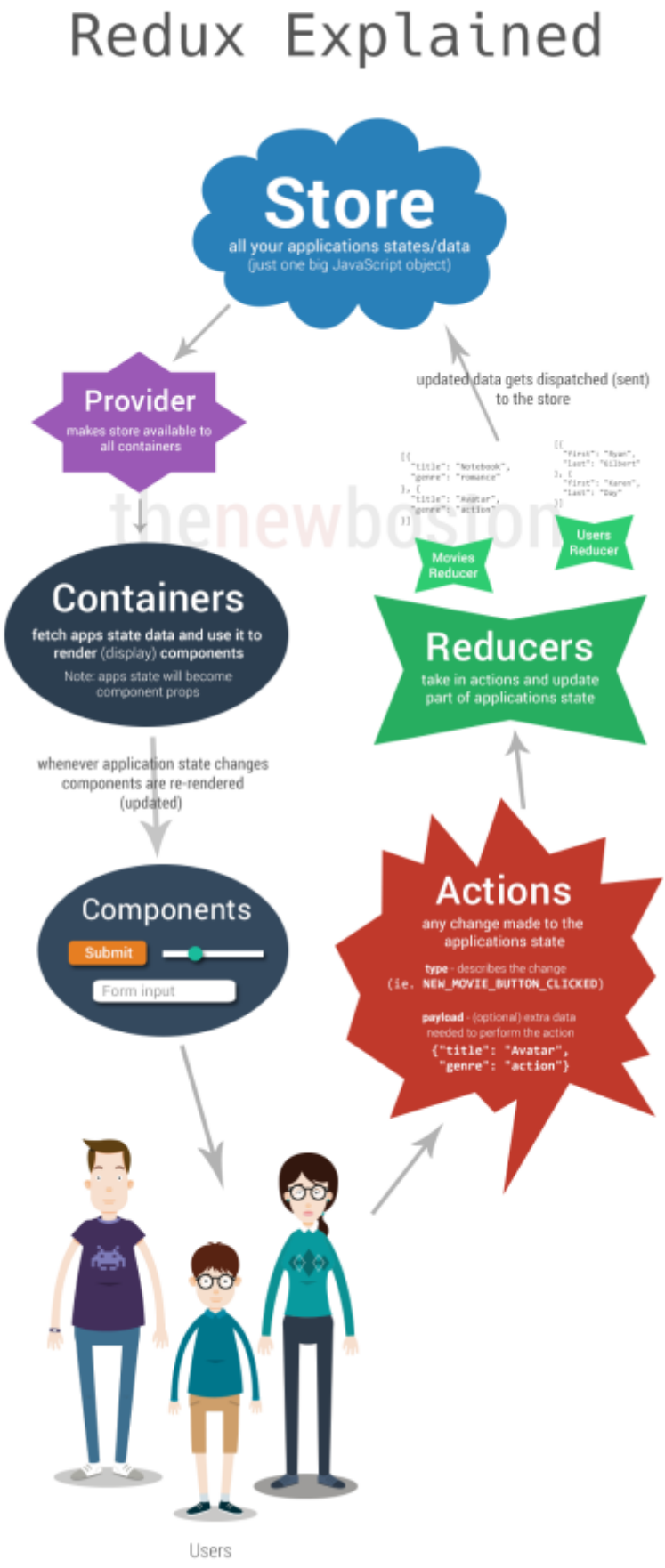

Figure 3: Redux structure suggestion [link]

All of the code is designed and written using ES6 modern style like:

- let, for introducing variables
- () => {…}, 'fat arrow' syntax for creating functions (Arrow functions)
- template literals, with snippets of JS inside string (backtick syntax), uncomment EndPoint triggers in src/index.js to test them
- …<array_name>, spread operators
- classes, and
- inheritance

Functional Requirements

At this point I am going to use three states: COMPLETED *for completed tasks*, CODE_EXISTS_NOT_WORKING *for tasks that weren't completed and code exists but is commented, the APIs with axios are working properly, the UI does not*, NOT_IMPLEMENTED *for tasks that I didn't have time to work on yet*

1. (COMPLETED) The interface shall be capable of displaying the products available on the system
    1.1 (COMPLETED) Products can be shown in a grid list or a simple table with action buttons
    1.2 (COMPLETED) Each product can be added to the current order
2 The interface shall be capable of creating an order with a list of products currently added to it
    2.1. (COMPLETED) The order list shall be a separate list containing all products added to the order
    2.2. (COMPLETED) The order list shall display the product quantity and price
    2.3. (COMPLETED) The order list shall display the total price

3. (COMPLETED) The interface shall allow the user to control the order
    3.1. (COMPLETED) The user can clear the order and start from the beginning
    3.2. (COMPLETED) The user shall be able to specify the customer name for this order
    3.3 (COMPLETED) The user can submit the order

4. The interface must allow users to view all orders in a separate page dedicated for order viewing
    4.1. (CODE_EXISTS_NOT_WORKING) Orders shall be displayed in a table
    4.2. (CODE_EXISTS_NOT_WORKING) Each order status can be changed by the user
    4.3. (CODE_EXISTS_NOT_WORKING) Orders can also be deleted by the user
    4.4. (NOT_IMPLEMENTED) Orders must be paginated to enhance user experience and lessen the load on the system.

5. The interface must display notification messages for any actions executed to the back-end system
    5.1. (CODE_EXISTS_NOT_WORKING) Success actions
    5.2. (CODE_EXISTS_NOT_WORKING) Errors generated by the back-end system

Setup

Step 1

Run the server you provided in the Appendix of the exercise:
$ java -jar exercise-pos-1.0-SNAPSHOT.jar

Step 2

Considering you have downloaded npm open terminal and run:
$ npm install
$ npm start

Step 3

Navigate to http://localhost:3000/
You are ready ☺

Credits

I had help and credits should be given to the people below for their great examples, tutorials and sharing of code (MIT licenced):

- Dan Abramov for redux examples
- Erik Rasmussen for the usage of ducks structuring specialization over react-redux
- Will Becker for this post explaining the best practices for all that I used
- Bucky Roberts for all the code and youtube tutorials for React and React Redux
- Kris Urbas for his repository over which I stepped to create this project, and
- Louis Barranqueiro for his great notification system reapop over react, that unfortunately I didn't have the time to make this work for me

Given the above I didn't got the time to create test for the API and UI testing. Of course, I believe that a good taste of my writing code skills is given though this project. Thank you a lot Silensec, for giving me the opportunity to prove myself, and I hope to be successful